IEEE Std 1609.2-2016 Guidance Note 7: Critical information fields

William Whyte, Security Innovation

Version 0.1, 2016-08-16

1 Introduction

This note identifies an area in IEEE Std 1609.2-2016 that has been reported as an incompleteness in the specification that may affect whether two implementations behave differently given a valid signed message received from a more recent implementation. The note gives guidance to implementers and certification testers to enable consistent implementations..

The defect concerns the treatment of fields that are not “understood” by an implementation of 1609.2, because they contain values that were defined in the standard after that implementation was created. (This can be done in 1609.2 without changing the version number of the structures due to the use of ASN.1 “extension markers”). In some cases 1609.2 defines whether a message containing an unrecognized field value should be considered valid or invalid: if a message should be rejected if a particular field value is unrecognized, that field is called a “critical information field”. However, 1609.2 does not give this guidance in all cases where it is possible a field value might be unrecognized. This means that some implementers might choose to accept a particular signed message, while others might choose to reject it.

This guidance note is structured as follows. Section 2 provides a summary of the issue and the potential impact. Section 3 reviews all the clauses of 1609.2-2016 where this issue might arise and recommends whether or not the standard should be modified in those places.

2 Summary of issue

2.1 ASN.1 extensibility

Extensibility has been a feature of ASN.1 since 1994. It allows designers of an ASN.1 module to identify that a structure might be extended, such that if optional new fields are added in subsequent versions, an encoding of the extended data structure without any of the fields present is identical to an encoding of the same structure in the original version. ASN.1 also provides mechanisms for designers to specify within the ASN.1 itself how unrecognized entries in a structure should be handled, but this appears to be little-used as exception handling is in general considered “application” behavior and out of the scope of the ASN.1 parser itself.

“ASN.1 Complete”, by John Larmouth, contains the following explanation of extensibility (Chapter 5, section 1)

What is "extensibility"? "Extensibility" refers to a combination of notational support, constraints on encoding rules, and implementation rules. This support enables a protocol specified (and implemented) as version 1 to be upgraded some years later to version 2 in specifically permitted ways. Provided the version 2 extensions are within the

permitted set of extensions (and provided the version 1 protocol was marked as "extensible"), then there will be a good interworking capability between the new version 2 systems and the already-deployed and unmodified version 1 systems.

The keys to extensibility are:

- To ensure that version 2 additions or extensions are "wrapped up" with length counts in encodings, and can be clearly identified by version 1 systems as "foreign material".
- To provide a clear specification that version 1 systems should process the parts of the encoding that are not "foreign material" in the normal version 1 way, and should take defined and predictable actions with the "foreign material".
- To avoid unnecessary (and verbose) wrappers and identifications in encodings by using notational "flags" on where version 2 additions or extensions may need to be made.

For the extensibility concept to be successful, all three of these components must be present.

Larmouth makes the following statement about when “extensibility behavior” is necessary (Chapter 7, Section 2.6.1):

It is absolutely vital that when you use ellipsis you give a clear statement of what behaviour you expect:

- From version 1 systems if they receive added material.
- How version 2 systems where mandatory fields have been added are to handle messages from version 1 systems.

The former is the more common case, as version 2 additions tend usually to be marked OPTIONAL.

… and identifies six main types of “extensibility behavior” (Chapter 7, Section 2.6.2.7):

- Silently ignore.
- Give some form of error response.
- Map to a version 1 value or object.
- Include a special "unknown value" in version 1 and specify its processing.
- Take the added material or unknown choice or value and relay it on unchanged.
- Process as much as possible then truncate (silently or with some form of error response).

Depending on the actual extensible construct, where that construct is used, the semantics associated with it, and how it affects later (perhaps much later) processing, we can choose one of these behaviours - or perhaps determine that another application-specific handling is more appropriate.

We can see that extensibility and correct behavior on “added” fields have been a concern of the ASN.1 and protocol design community for a long time, and also that there is no single convention for how added fields should be handled. The approach in 1609.2 – to specify how to handle unrecognized fields in the text description of each structure – is consistent with this overall recognition that added field handling is setting-specific.

2.2 Terminology: “parse” v “interpret”

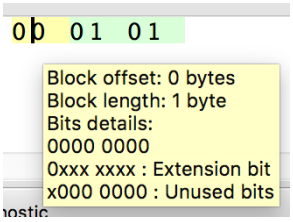
It's best to illustrate "parse" v "interpret" with an example. Consider the following version 1 ASN.1 structure, which includes an extensibility marker:

```
MyExtType ::= SEQUENCE {
   a INTEGER,
   ...
}
```

Then say that at some point in the future the version 2 structure is extended as follows:

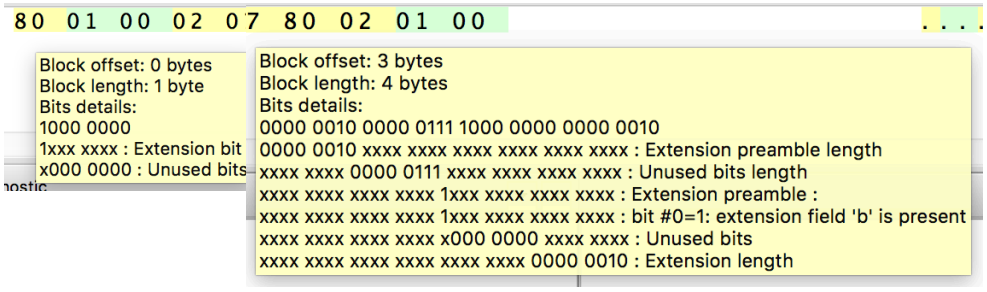
```
MyExtType ::= SEQUENCE {
   a INTEGER,
   ...,
   b INTEGER
}
```

A Canonical-OER encoding of a=0, with b omitted, looks like this (screenshot thanks to OSS Nokalva):



As can be seen, the first bit of the first byte of the encoded structure, known as the preamble tag, is used to indicate whether the structure has been extended. (Extensible structures therefore always take at least one more byte to encode than an unextensible equivalent would, even if none of the extension fields are present). Since no extension fields are present, this structure looks the same whether the sender is using version 1 or version 2 of the structure definitions.

A Canonical-OER encoding of a = 0, b = 0, looks like the following (screenshot thanks again to OSS Nokalva):



The first yellow block is the preamble tag, the second yellow block is the extension tag. The extension tag encodes the following information:

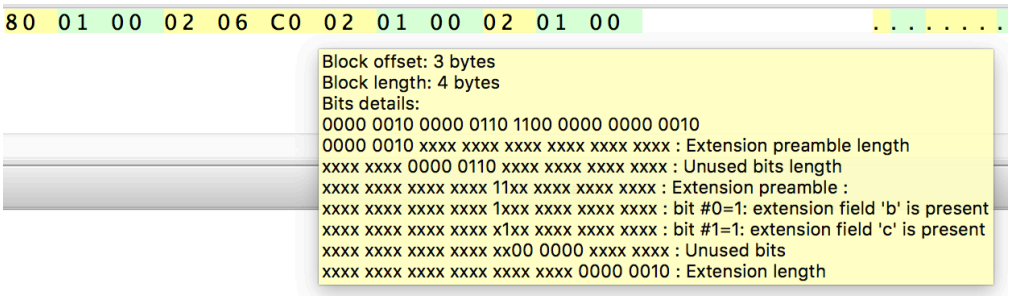
- The first byte indicates that there are two bytes of “extension preamble”, and therefore the total length of the extension tag is four bytes.
- The second and third byte indicate that there is one extension present, and that the total possible number of extensions is one.
- The fourth byte indicates that the length of the first extension following is 2 bytes.

The extension contains the INTEGER b of length 1 and value 0.

If at some further point in the future the version 3 structure were to be defined as follows:

```
MyExtType ::= SEQUENCE {
   a INTEGER,
   ...,
   b INTEGER,
   c INTEGER
}
```

Then an encoding of a = 0, b = 0, c = 0 would look like the following:



Note the change in the “unused bits length” to 6 and the change in the bitmask in the extension preamble to indicate that both b and c are present. Note also that each extension begins with a length field of its own, and then contains an encoding of the INTEGER type that contains the length and value of the INTEGER.

We are finally at a point where we can explain “parse” v “interpret”.

An implementation can *parse* a structure if it can identify where the fields in the structure begin and end, and identify the end of the structure to parse any structure that comes after it in the received encoded data. **Any implementation of any version of an extensible structure can parse the structure**.

An implementation can *interpret* the structure if it can parse it and also can map each encoded field to a known entry in the ASN.1 structure. For example, a parser for version 1 of MyExtType could not interpret an encoding containing a=0, b=0, because it would not know that b was an integer. **An implementation can interpret a structure only if it implements a version of the structure in which all the extension fields present in the structure are defined**.

2.3 Critical information fields in 1609.2

The concept of "critical information fields" is introduced in 1609.2 clause 5.2.5 as follows, to cover fields that the Secure Data Services can parse but not interpret (note that the word "parse" is unfortunately used in the sense "interpret" in the quoted paragraph):

> Critical information fields are any fields necessary to establish the validity of a signed SPDU. An implementation of WAVE Security Services that cannot parse critical information fields in a signed SPDU or a certificate shall consider that signed SPDU or certificate to be invalid.
>
> An implementation of WAVE Security Services might not be able to parse critical information fields for a number of reasons, including:
>
> — The fields are too long.
>
> — An array contains too many entries.
>
> — A recursive structure contains too many recursions.
>
> — A structure that uses identifiers includes an identifier that the implementation does not recognize.
>
> For each data type defined in Clause 6 that may be of arbitrary length (in octets or number of entries) or depth, the definition in Clause 6 specifies the circumstances under which it is a critical information field, and a minimum size to be supported by any conformant implementation of WAVE Security Services. The Protocol Implementation Conformance Statement (PICS) provided in Annex A allows an implementation to state any size it supports beyond the minimum required for conformance.

As stated in 5.2.5, 1609.2 does provide guidance in all the cases above. However, it does not provide guidance in every case where the structure contains extension fields. This means that two different v3 implementations might behave differently when they encounter an "added field" inserted by an implementation of v4 (or some other future version).

2.4 Why accept? Why reject?

There are two types of messages that might be rejected:

- Signed messages might not pass verification
- Encrypted messages might not successfully decrypt.

A signed message should be rejected if the information that cannot be parsed might make a difference to whether or not the message can be trusted. This might be for one of two reasons:

- Failure to parse the information literally means the receive-side implementation doesn't have enough information to complete the verification process. For example, the signature might be for an unknown algorithm, or the signer identifier might be of an unknown type.
- There are fields in the signed message that might in principle carry trust information and the receiver does not understand their meaning well enough to decide whether they do in fact carry trust information. Therefore, the receiver cannot definitively determine that the message should be trusted.

An encrypted message should be rejected only if the receiver cannot decrypt it: unparseable fields should matter only if they directly impact decryption. For example, if the symmetric encryption algorithm is not recognized, the receiver cannot decrypt the message. However, if the message was encrypted for multiple recipients, and the receiver cannot interpret information intended for one of the other recipients, this should not result in the message being rejected as this failure to interpret does not affect the receiver's ability to decrypt.

2.5 Issue and impact

Some structures in 1609.2 contain extensibility markers and do not provide guidance as to what to do on failure to interpret fields after the extension marker. Other structures contain guidance that on review appears to be wrong by the principles outlined in section 2.4 of this paper. Finally, the structure RegionAndSubregions, contain a field that is only defined if the enclosing CountryAndSubregions structure indicates that the country is the USA.

Impact. None of these affect interoperability, as there are no extensions or undefined values of fields in use at the moment. All of these cases should be clarified, though, to prevent future inconsistencies

3 Relevant sections of 1609.2

This section of this document reviews the relevant sections of 1609.2 to identify all sections that might not be interpretable by older implementations, review the current recommendations, and recommend changes if the current recommendations do not meet the principles outlined in section 2.4 of this paper.

This section reviews changes in the main body of 1609.2. Shading means the following:

- Green: No change needed
- Blue: Add a statement that there are no critical information fields
- Orange: Add a statement that there are critical information fields
- Red: Existing statement is incorrect and should be changed

Although these changes could in principle have affected the Protocol Implementation Conformance Statement (PICS) in Annex A, we believe on review that no changes to the PICS are necessary.

Clause number and name	Current status	Recommended action	Rationale
5.2.5	Uses "parse" to mean what we mean by "interpret" in this document	Change "parse" to "interpret"	Elsewhere in the standard "parse" is always used in the sense it's used in this document, so it would be good to be consistent.

6.3.3 Ieee1609-Dot2Content	Extensible, no guidance given	Add statement that there are no critical information fields	Ieee1609dot2Data is a top-level enclosing structure and decisions to accept/reject are not made at this level.
6.3.5 Hash-Algorithm	Extensible, critical	No change	A receiver who can't interpret this doesn't know which hash algorithm to use.
6.3.7 Signed-DataPayload	Extensible, not critical	No change	Payload is interpreted by the SDEE, not the security services.
6.3.8 Hashed-Data	Extensible, critical	No change	A receiver who can't interpret this doesn't know which hash algorithm to use.
6.3.9 Header-Info	Extensible, no guidance given	Add statement that there are no critical information fields	Assumption is that additional HeaderInfo fields are used for security management like p2p cert distribution, not to establish trustworthiness
6.3.16 MissingCrl-Identifier	Extensible, no guidance given	Add statement that there are no critical information fields	Not used by current implementations but defined for future use; non-critical because it does not help establish the trustworthiness of the message it is contained in.
6.3.19 Symmetric-Encryption-Key	Extensible, critical in signed message	Should not be critical	In a signed message, this field is used to encrypt responses to the message or certificate that contains it. Even if it cannot be interpreted, it has no bearing on the trustworthiness of the sender.
6.3.21 Symm-Algorithm	Extensible, critical in signed message	Should not be critical	In a signed message, this field is used to encrypt responses to the message or certificate that contains it. Even if it cannot be interpreted, it has no bearing on the trustworthiness of the sender.
6.3.21 Base-Public-Encryption-Key	Extensible, critical in signed message	Should not be critical	In a signed message, this field is used to encrypt responses to the message or certificate that contains it. Even if it cannot be interpreted, it has no bearing on the trustworthiness of the sender.
6.3.24 Signer-Identifier	Extensible, critical	No change	If it can't be interpreted, the receiver doesn't know what key to use to verify the signed message.
6.3.28	Extensible, no	Should be critical	If it can't be interpreted, the receiver

Signature	guidance given		can't interpret the signature and so can't verify.
6.3.35 Encrypted-Data-Encryption-Key	Extensible, critical	Should only be critical if in the receiver's RecipientInfo	If it's in the receiver's RecipientInfo and can't be interpreted, the receiver can't decrypt. If it's in some other receiver's RecipientInfo it has no bearing on the receiver's ability to decrypt.
6.3.37 Symmetric-Ciphertext	Extensible, critical	No change	If the receiver can't identify the symmetric algorithm, they can't decrypt.
6.4.4 Certificate-Type	Extensible, no guidance given	Should be critical	If value isn't recognized the certificate can't be parsed correctly.
6.4.7 Issuer-Identifier	Extensible, no guidance given	Should be critical	If value isn't recognized the certificate can't be parsed correctly.
6.4.8 ToBe-Signed-Certificate	Extensible, no guidance given on unrecognized extension fields	Should be critical	The assumption is that in general fields in the certificate are there to establish trustworthiness and so should be critical.
6.4.9 CertificateId	Extensible, critical	No change	Fields in the certificate are there to establish trustworthiness and so should be critical. A plausible use for a new version of this field would be to support another means of revocation; if the receiving device could not interpret this field it could not tell whether the sender was revoked.
6.4.17 Geographic-Region	Extensible, critical	No change	If value isn't recognized the cert and message validity can't be established.
6.4.22 Identified-Region	Extensible, critical	No change	If value isn't recognized the cert and message validity can't be established.
6.4.26 RegionAnd-Subregions	No guidance given	Should be critical	The subregions field is explicitly not defined for regions outside the US, so cannot be recognized by implementations of this version of the standard. If value isn't recognized the cert and message validity can't be established.
6.4.29 Service-Specific-	Extensible, critical	Should not be critical	SSP is not consumed by the security services but passed up to the SDEE to consume. As such, the application can

Permissions			decide to reject a message on the basis of an uninterpretable or unparseable SSP and the security services should not have to. **The specification of the interface needs to make it clear that the SSP returned to the SDEE must include an indication of which CHOICE was made for the SSP rather than just returning the contents, as this CHOICE information is vital to the interpretation made by the receiving SDEE**.
6.4.31 Subject-Permissions	Extensible, critical	No change	If value isn't recognized the cert and message validity can't be established.
6.4.34 Subject-Permissions	Extensible, critical	No change	If value isn't recognized the cert and message validity can't be established.
6.4.35 Verification-KeyIndicator	No guidance given	Should be critical	If value isn't recognized the cert and message validity can't be established.
6.4.36 Public-Verification-Key	Extensible, critical	No change	If value isn't recognized the cert and message validity can't be established.